Asymptotic Determinacy of Path Queries using Union-of-Paths Views

Asymptotic Determinacy of Path Queries using Union-of-Paths Views

Nadime Francis

ENS-Cachan, Inria
francis@lsv.ens-cachan.fr

Abstract

We consider the view determinacy problem over graph databases for queries defined as (possibly infinite) unions of path queries. These queries select pairs of nodes in a graph that are connected through a path whose length falls in a given set. A view specification is a set of such queries. We say that a view specification $\mathbf{V}$ determines a query Q if, for all databases D, the answers to $\mathbf{V}$ on D contain enough information to answer Q.

Our main result states that, given a view $\mathbf{V}$, there exists an explicit bound that depends on $\mathbf{V}$ such that we can decide the determinacy problem for all queries that ask for a path longer than this bound, and provide first-order rewritings for the queries that are determined. We call this notion asymptotic determinacy. As a corollary, we can also compute the set of almost all path queries that are determined by $\mathbf{V}$.

1998 ACM Subject Classification H.2.4 [Database Management]: Systems – *Query processing*, H.2.3 [Database Management]: Languages – Query languages

Keywords and phrases Graph databases, Views, Determinacy, Rewriting, Path queries

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.44

1 Introduction

View determinacy is a static analysis problem on databases that consists in deciding whether a given set of initial queries, called a view, contains enough information to answer a new query, and this on all databases. Solving this problem has many applications, namely in query optimization and caching. Assume that querying the database is costly, but that answers to all previous queries are kept in cache. Then it is useful to know whether a new query can be answered using only cached information and without accessing the database. Query determinacy can also be stated as a security problem. Assume that views represent information that can be publicly accessed, but that the considered query contains private data that should not be disclosed. Then it should be ensured that the view does not determine the query.

We consider this question over graph databases. Graph databases are relational databases in which all relations are binary. Equivalently, they can be seen as directed graphs with edges labeled from a finite alphabet. Such databases arise naturally in several scenarios, which include social networks, crime detection, biological data and the semantic Web. For instance, in social networks, individual data such as name or phone number are represented as nodes, whereas relationships between members of the network are edges linking the corresponding nodes and labeled by the nature of the relationship. Thus, a person X is a friend of a person Y if there is an edge going from X to Y with label *friend*.

Information contained in a graph database does not only lie in the content of the graph but also in its topology, that is in *how* the different data nodes are connected to each other. Typical queries then naturally ask about topological properties of the graph, namely the

18th International Conference on Database Theory (ICDT'15).
Editors: Marcelo Arenas and Martín Ugarte; pp. 44–59

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

existence of links, paths, and so on. In a social network, a user X could be interested in computing the transitive closure of the *friend* relation: she would like to retrieve all nodes Y such that there is a path going from X to Y using only the *friend* label. Relevant query and view languages to consider in this context should have at least this expressive power.

The determinacy problem has been considered in various contexts (see [1], [5] among others). It was shown in [2] that determinacy is decidable when queries and views are defined as path queries, that is, queries Q_k that select pairs of nodes (x, y) such that there is a path from x to y whose length is a given integer k. For instance, it proves that the two views Q_3 and Q_4 determine the query Q_5, which is not immediate to see. The main contribution of our work is to extend this result by considering a broader class of views that allows disjunction. For instance, a query $Q_{k,\ell}$ selects pairs of node that are linked either by a path of length k or a path of length ℓ. A typical case covered by our work is the following: views are Q_2, $\mathrm{Q}_{1,2}$ and $\mathrm{Q}_{2,3}$, and we will see that these views determine the query Q_5.

More precisely, we consider here arbitrary unions of path queries. A union of path queries Q is a query that selects pairs of nodes in a graph that are connected through a path whose length falls in a given set. Note that we do not have any restriction on how these sets are defined, in particular we do not require them to be finite, or even have a finite representation. Of course, our algorithmic constructions only work when these representations are effective, but our theoretical criteria do not require it. We actually require very little of these representations, and a lot of formalisms would fit our needs. For instance, regular path queries on a one letter alphabet could define the query Q_{odd} that selects all pairs of nodes linked by a path of odd length, and a query similarly defined by a context-sensitive language could be Q_{exp}, that selects pairs of nodes linked by a path of length 2^n for some n. Note that a path query can be seen as a special case of union of paths queries of size 1.

In this paper, we show that, given a view $\mathbf{V}$ defined by unions of path queries, we can decide whether $\mathbf{V}$ determines a path query Q assuming Q is "big enough," that is, Q asks for the existence of a path longer than some n_0 that we can effectively compute from $\mathbf{V}$. We call this notion *asymptotic determinacy*. Although n_0 is of exponential size, our decision procedure actually works in Π_2^{P} in the size of the finite sets that are associated with the view, disregarding the infinite ones. When it concludes that $\mathbf{V}$ determines Q, we also provide a first-order rewriting of Q using $\mathbf{V}$. Otherwise, it produces a generic counter-example that shows that $\mathbf{V}$ does not determine Q. Our technique starts by reducing $\mathbf{V}$ to a much simpler view $\mathbf{V}'$ that has many useful properties, namely all queries in $\mathbf{V}'$ are finite unions, and some $Q \in \mathbf{V}'$ is actually a path query Q_c, for some integer c. This particular query is a key to our reasoning, as it allows us to reduce infinite structures to finite ones by computing modulo c. The finite number of small queries that we are not able to process are cases where both our criterion of determinacy and our generic counter-examples fail.

Related Work

The determinacy problem has been considered in [3] for regular path queries, i.e. queries that select pairs of nodes that are connected through a path whose sequence of labels satisfies some regular expression. In [3], determinacy is known as *losslessness under the exact views assumption*. However, it is still unknown whether this property is decidable and what a good rewriting language could be. Note that, on a one-letter alphabet, regular path queries are actually weaker than infinite unions of path queries.

The determinacy problem has been solved in two specific cases. First, [2] showed how to decide whether a path view determines a path query and provides first-order rewritings of the query using the view when it is the case. The work presented here can be seen as

an extension of [2], where we consider more expressive views, by allowing (possibly infinite) disjunction. In [2], a simple criterion is given: the view determines the query if and only if the view image of a simple path satisfying the query is connected. We will see in Example 14 and Example 15 that this decision criterion no longer applies here, as the view images in these examples are connected, but the view does not determine the query.

Secondly, [4] proved that regular path queries can always be rewritten as Datalog queries using regular path views, assuming *monotone determinacy*. Monotone determinacy is a stronger form of determinacy that is known to be decidable in this setting. It basically requires both determinacy and the fact that rewritings of the query using the views are monotone. Our work can be seen as an attempt to lift this monotonicity assumption, while still retaining some of the expressive power of regular path queries and views, such as disjunction and transitive closure. Of course, without assuming monotonicity there can be no hope of finding a rewriting in Datalog, since it can only express monotone queries.

2 Preliminaries

Graph Databases

A *binary schema* σ is a finite set of binary relational predicates. A *graph database* D over σ is a relational structure over a binary schema σ. Alternatively, it can also be seen as a directed edge-labeled graph whose labeling alphabet are symbols in σ. An element in the domain of D is called a *node*.

A *path* π in a database D from x_0 to x_m is a finite sequence $\pi = x_0 a_0 x_1 \dots x_{m-1} a_{m-1} x_m$, where each x_i is a node of D, each a_i is a symbol of σ, and for all i, $a_i(x_i, x_{i+1})$ holds in D. Such a path is said to be *simple* if each node occurs at most once in π. We simply write $\pi = x_0 \dots x_m$ when the specific symbol of σ that holds for each pair $(i, i+1)$ is irrelevant. The length of π, denoted by $|\pi|$, is the length of the word $a_0 \dots a_{m-1}$, in this case m. To denote the fact that π is a path from x_0 to x_m, we will often write $x_0 \xrightarrow{\pi} x_m$. By abuse of notation, we also consider π as a graph database that contains exactly the nodes $x_0, \dots, x_m$, and in which only $a_i(x_i, x_{i+1})$ holds, for all i.

Queries

A binary *query* Q over a schema σ is a mapping associating to each graph database D over σ a binary relation Q(D) over the domain of D. In this work, we only consider the following two query languages.

A *path query* Q is defined by a single integer k. On a given database D, Q returns all the pairs of nodes (x, y) in D such that there exists a path in D from x to y of length k. In other words, $\mathrm{Q(D)} = \{(x, y) \in \mathrm{D} \mid \exists \pi, x \xrightarrow{\pi} y \text{ and } |\pi| = k\}$. For ease of notation and consistency with what follows, we will write $\mathrm{Q} = \{k\}$.

A *union of path queries* Q is defined by a (possibly infinite) set of integers $\{k_1, k_2, \dots\}$, and returns all the pairs of nodes that are connected through a path whose length belongs to the set, i.e. $\mathrm{Q(D)} = \{(x, y) \in \mathrm{D} \mid \exists \pi, x \xrightarrow{\pi} y \text{ and } |\pi| \in \mathrm{Q}\}$. By abuse of notation, Q represents both the query and the associated set. Unions of path queries are a generalization of path queries, as any path query can be seen as a union of path queries whose associated set is of size 1.

These two query languages can be compared with other core languages commonly used in the field. Path queries are conjunctive queries on a single predicate whose underlying graph is a directed path. Alternatively, they are also regular path queries whose associated

language is a single word on a single-letter alphabet. Unions of path queries are arbitrary unions of such queries. Note that these are more expressive than regular path queries on a single letter alphabet. Indeed, $Q = \{p \mid p \text{ is prime}\}$ is a union of path queries that is not regular.

We do not impose any way of representing the infinite sets associated to unions of path queries. For our constructions to be effective, we only require:

- the ability to decide, given a query, whether its associated set is infinite.
- the ability to effectively list all the elements in the associated set, when it is finite.

Thus, many formalisms would fit our needs, such as regular sets, but we could possibly think of stronger languages for finitely describing infinite sets of integers. While considering infinite unions may seem rather strange, this should be understood on a conceptual level, as a way to ease comparisons and extensions to existing work. Most of the work presented is actually only relevant to finite unions. Indeed, we will see in Lemma 4 that infinite unions cannot be used for the determinacy and rewriting of path queries, which explains the very low requirements we have on queries with an infinite associated set.

Views

Let σ and τ be two binary schemas. A *view* $\mathbf{V}$ from σ to τ is a set of binary queries over σ, one for each symbol in τ. Note that, since both σ and τ are finite, then $\mathbf{V}$ is also a finite set. By abuse of notation, we will use the same notation for both the relational predicate in τ and the corresponding query in $\mathbf{V}$. For a given graph database D over σ, $\mathbf{V}(\mathrm{D})$ is then defined as another graph database E over τ, such that for each $V \in \tau$, the interpretation of V in E is exactly $V(\mathrm{D})$. Finally, the nodes of E are exactly those that appear in one of those relations, also known as the *active domain* of E.

Determinacy

A formal definition of determinacy is given in [6] as:

▶ **Definition 1** (Determinacy). We say that a view $\mathbf{V}$ determines a query Q if:

$$\forall \mathrm{D}, \mathrm{D}', \mathbf{V}(\mathrm{D}) = \mathbf{V}(\mathrm{D}') \;\Rightarrow\; \mathrm{Q}(\mathrm{D}) = \mathrm{Q}(\mathrm{D}')$$

Intuitively, this means that a view $\mathbf{V}$ determines a query Q, which we write $\mathbf{V} \twoheadrightarrow \mathrm{Q}$, if, for all databases D, $\mathbf{V}(\mathrm{D})$ always contains enough information to answer Q on D. Moreover, we say that a query R is a *rewriting* of Q using $\mathbf{V}$ if $\mathrm{R}(\mathbf{V}(\mathrm{D})) = \mathrm{Q}(\mathrm{D})$ for all D.

The *determinacy problem* is the problem of deciding, given a view $\mathbf{V}$ and a query Q, whether $\mathbf{V} \twoheadrightarrow \mathrm{Q}$. To the best of our knowledge, its decidability status is still open when Q is a conjunctive query and $\mathbf{V}$ a conjunctive view, and also when Q is a regular path query, and $\mathbf{V}$ a regular path view. Nonetheless, several cases have been considered and solved. The results in [2] solve the problem when Q is a path query and $\mathbf{V}$ a path view. In [6], this problem is considered for Q and $\mathbf{V}$ defined by conjunctive queries, and in [4] for Q and $\mathbf{V}$ defined with regular path queries, both with the added restriction that $\mathbf{V}$ must determine Q in a monotone way, which means that a monotone rewriting of Q using $\mathbf{V}$ exists.

Here, we consider the determinacy problem for Q defined as a path query, and $\mathbf{V}$ defined as a set of unions of path queries. However, for each $\mathbf{V}$, there exist a finite number of Q on which our technique does not work. Hence, we are actually solving a slightly weaker problem, that we call the *α-asymptotic determinacy problem* which allows, for each $\mathbf{V}$, to exclude a finite number of queries Q. The excluded queries are those that ask for a path longer than

$\alpha(\mathbf{V})$, where α is a fixed function that maps each view to a natural number. Note that providing an answer or a rewriting in "almost all" cases is something that has already been considered in the same context, for instance in [2]. We can now formally state the problem and our main result:

PROBLEM :	α-ASYMPTOTIC DETERMINACY
INPUT :	A union-of-paths view $\mathbf{V}$ and a path query $\mathrm{Q} = \{n\}$ with $n > \alpha(\mathbf{V})$
QUESTION :	Does $\mathbf{V} \twoheadrightarrow \mathrm{Q}$?

▶ **Theorem 2.** *There exists an explicit and computable function α for which the α-asymptotic determinacy problem is decidable. Moreover, when the view determines the query, the decision procedure effectively computes a first-order rewriting of the query using the view.*

It will actually come from the proof that the specific α for which we can solve the problem grows exponentially in the size of the finite unions in $\mathbf{V}$, while disregarding the infinite ones. However, the decision procedure itself works with a much lower Π_2^{P} complexity.

Arithmetic Notations

Some of the proofs in this work involve a lot of arithmetic reasonings. We present here the notations that we use. Given two integers n and d, $n[d]$ represents the remainder in the division of n by d. We say that two integers n_1 and n_2 are equivalent modulo d, and we write $n_1 \equiv n_2[d]$ if they have the same remainder modulo d. We denote by $\gcd(A)$ the greatest common divisor of a set of integers A, and we use $n_1 \wedge n_2$ for $\gcd(\{n_1, n_2\})$. Additionally, for two binary relations R and S, we write $R \cdot S$ for $\{(x, z) \mid \exists y, R(x, y) \text{ and } S(y, z)\}$. Let n be a positive integer, we also use $R^1 = R$, and $R^n = R^{n-1} \cdot R$ if $n \geq 2$.

Organization

The rest of the paper investigates the determinacy problem for a path query Q using view $\mathbf{V}$ defined by unions of path queries. In Section 3, we start by providing some conditions on Q and $\mathbf{V}$ that are necessary for $\mathbf{V}$ to determine Q. Section 4 is dedicated to proving Theorem 2, which gives a procedure for deciding the determinacy problem for almost all path queries Q, as well as a first-order rewriting of Q using $\mathbf{V}$ for the queries that are determined. Finally, in Section 5, we discuss the issue that remains to be solved in order to decide determinacy for all queries.

Note that, due to space constraints, we only provide sketches of the most long and technical proofs. The missing proofs can be found in a more complete version of this paper.

3 Necessary Conditions and First Results

In this section and the next, we only consider path queries and union of path views. When we simply say "query" or "view", it is implied that they belong to those specific classes. The goal here is to provide a set of necessary conditions for a view $\mathbf{V}$ to determine a query Q.

Our first lemma states that a view $\mathbf{V}$ cannot possibly determine a query Q if $\mathbf{V}$ does not at least contain a path query. In other words, even though $\mathbf{V}$ is defined using union of path queries, at least one of them cannot make use of the union.

▶ **Lemma 3.** *Assume that a view $\boldsymbol{V}$ and query Q are such that $\boldsymbol{V} \twoheadrightarrow Q$. Then there exists $C \in \boldsymbol{V}$ such that $|C| = 1$.*

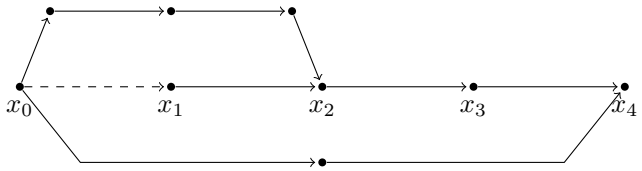

Figure 1 Illustration for the proof of Lemma 3, showing here that $V = \{2, 4\} \not\twoheadrightarrow \mathrm{Q} = \{4\}$. Following the notations in the proof, the top path is $\pi_{2,V}$ and the bottom path is $\pi_{4,V}$. Remark then that adding or removing the dashed edge does not change the view, but changes the query result.

Proof. Assume by contraposition that, for all $V \in \mathbf{V}$, $|V| > 1$. Let $\mathrm{Q} = \{n\}$. We build a database D as follows:

- D contains $n + 1$ distinct nodes $x_0, \ldots, x_n$.
- For all $i < n$, $a(x_i, x_{i+1})$ holds in D.
- For all $i \leq n$, for all $V \in \mathbf{V}$ such that $i \in V$, we add to D a simple path $\pi_{i,V}$ from x_0 to x_i, such that $|\pi_{i,V}| \in V - \{i\}$. Such a path exists because $|V| > 1$.

We then construct another database D' which is a copy of D except that $a(x_0, x_1)$ does not hold in D'. It is then easy to check that $\mathbf{V}(\mathrm{D}) = \mathbf{V}(\mathrm{D}')$ and that $\mathrm{Q}(\mathrm{D}) \neq \mathrm{Q}(\mathrm{D}')$. In particular, $(x_0, x_n) \in \mathrm{Q}(\mathrm{D})$ and $(x_0, x_n) \notin \mathrm{Q}(\mathrm{D}')$. Hence $\mathbf{V} \not\twoheadrightarrow \mathrm{Q}$, which concludes the proof. This construction is illustrated on Figure 1. ◀

Our second lemma states that we can safely ignore the queries in $\mathbf{V}$ that are defined by infinite unions. This means that if a view $\mathbf{V}$ determines a query Q, then $\mathbf{V}$ also determines Q without making use of its infinite components.

▶ **Lemma 4.** *Let Q be a query and $\boldsymbol{V}$ be a view. Let $\boldsymbol{V} = \boldsymbol{V}_f \uplus \boldsymbol{V}_\infty$, such that $\boldsymbol{V}_f$ only contains queries defined by finite sets, and $\boldsymbol{V}_\infty$ only contains queries defined by infinite sets. Then $\boldsymbol{V} \twoheadrightarrow Q$ if and only if $\boldsymbol{V}_f \twoheadrightarrow Q$.*

Proof. It is easy to see that if $\mathbf{V}_f \twoheadrightarrow \mathrm{Q}$, then $\mathbf{V} \twoheadrightarrow \mathrm{Q}$. Conversely, assume that $\mathbf{V}_f$ does not determine Q. Then there exists two databases D_1 and D_2 such that D_1 and D_2 agree on $\mathbf{V}_f$ but not on Q. Let k be the biggest number that appears in $\mathrm{Q} \cup \mathbf{V}_f$. We transform D_1 into a new database D'_1 as follows:

- We add to D'_1 $k + 1$ new nodes $x_0, \ldots, x_k$, as well as the following edges:
  - For all i, $a(x_i, x_{i+1})$ holds in D'_1.
  - $a(x_0, x_0)$ and $a(x_k, x_k)$ hold in D'_1.
- For each original node x of D_1, we add $a(x, x_0)$ and $a(x_k, x)$ to D'_1.

We then apply the same steps to D_2 and get a new database D'_2. This construction has no effect on Q or $\mathbf{V}_f$ for the original nodes of D_1 and D_2. However, it makes it so that for each $(x, y) \in \mathrm{D}_1$ (respectively D_2), for each $V \in \mathbf{V}_\infty$, $V(x, y)$ holds in $\mathbf{V}_\infty(\mathrm{D}'_1)$ (respectively $\mathbf{V}_\infty(\mathrm{D}'_2)$). Thus, we can check that D'_1 and D'_2 agree on $\mathbf{V}$ but not on Q. Hence $\mathbf{V} \not\twoheadrightarrow \mathrm{Q}$, which concludes the proof. ◀

Altogether, these two lemmas show than we can restrict our attention to views $\mathbf{V}$ that contain only queries defined by finite sets, and contain at least one query C such that $|C| = 1$. This reduction is effective if we can decide which views correspond to infinite sets, and which views correspond to singletons. We can now state the following definition for views that contain such a path query C:

▶ **Definition 5** (Complete)**.** Let $\mathbf{V}$ be a view and $C \in \mathbf{V}$ such that $C = \{c\}$. We say that $\mathbf{V}$ is *C-complete* if, for all $i \in \{0, \ldots, c - 1\}$, there exists $V \in \mathbf{V}$ and $k \in V$ such that $k \equiv i[c]$.

Our next necessary condition is an adaptation of the condition in [2]:

▶ **Claim 6.** *Let $\boldsymbol{V}$ be a view and $Q = \{n\}$. Let $\pi = x_0 \dots x_n$. If $\boldsymbol{V} \twoheadrightarrow Q$ then there is an undirected path from x_0 to x_n in $\boldsymbol{V}(\pi)$.*

This condition allows us to reduce any determinacy problem to an equivalent problem with the added hypothesis that the view is C-complete:

▶ **Lemma 7.** *Let $\boldsymbol{V}$ be a view and $C \in \boldsymbol{V}$ such that $C = \{c\}$. Let $Q = \{n\}$, and $\pi = x_0 \dots x_n$. Assume that there is a path from x_0 to x_n in $\boldsymbol{V}(\pi)$. Then we can effectively compute a view $\boldsymbol{V}'$ with $C' \in \boldsymbol{V}'$ such that $C' = \{c'\}$ and a query Q' such that $\boldsymbol{V}'$ is C'-complete and $\boldsymbol{V} \twoheadrightarrow Q$ if and only if $\boldsymbol{V}' \twoheadrightarrow Q'$.*

Sketch of proof. There are two cases to this proof. Consider the set U of all numbers that appear in $\mathbf{V}$. If $\gcd(U) = 1$, then we can construct a new query V as a combination of the other queries in $\mathbf{V}$, such that V contains some $v_i \equiv i[c]$ for all i.

Assume now that $\gcd(U) = d$, with $d \neq 1$. This means that all numbers appearing in $\mathbf{V}$ can be divided by d. Since x_0 is connected to x_n in $\mathbf{V}(\pi)$, this also means that d divides n. In this case, we build a new view $\mathbf{V}'$ and a new query Q' by dividing by d all numbers that appear respectively in $\mathbf{V}$ and Q. We then show that $\mathbf{V}$ determines Q if and only if $\mathbf{V}'$ determines Q', and we apply the first case to $\mathbf{V}'$ and Q'. ◀

Finally, the last lemma of this section shows that proving that $\mathbf{V} \twoheadrightarrow \mathrm{Q}$ for some Q also yields a lot of other determinacy results easily.

▶ **Lemma 8.** *Let $\boldsymbol{V}$ be a view and $C \in \boldsymbol{V}$ such that $C = \{c\}$. Let $Q = \{n\}$, and assume that $\boldsymbol{V} \twoheadrightarrow Q$. Then, for all positive integer k, $\boldsymbol{V} \twoheadrightarrow \{n + kc\}$.*

Proof. Let R be a rewriting of Q using $\mathbf{V}$. Let k be a positive integer. Then it is easy to check that $R \cdot C^k$ is a rewriting of $\mathrm{Q}' = \{n + kc\}$ using $\mathbf{V}$. ◀

4 Asymptotic Determinacy

The goal of this section is to prove Theorem 2. By using the results of Section 3, we can restrict our attention to C-complete views $\mathbf{V}$, with $C \in \mathbf{V}$ and such that all $V \in \mathbf{V}$ are finite. Theorem 2 is a consequence of the following proposition:

▶ **Proposition 9.** *Given a C-complete view $\boldsymbol{V}$ defined by finite unions of path queries, such that $C \in \boldsymbol{V}$ with $C = \{c\}$ for some $c \in \mathbb{N}$ and a natural number $o \in \{0, \dots, c-1\}$, it is decidable whether there exists a query $Q = \{n\}$ such that $n \equiv o[c]$ and $\boldsymbol{V} \twoheadrightarrow Q$. If this is the case, such a query Q and a first-order rewriting of Q with regards to $\boldsymbol{V}$ can be effectively computed.*

Indeed, given a C-complete view $\mathbf{V}$ and a query $\mathrm{Q} = \{m\}$, we can first decide if there exists another query n, with $n \equiv m[c]$ such that $\mathbf{V} \twoheadrightarrow \{n\}$. If this is not the case, then we can safely conclude that $\mathbf{V} \not\twoheadrightarrow \mathrm{Q}$. Otherwise, Proposition 9 gives us an explicit n that is determined by $\mathbf{V}$. If $m > n$, then Lemma 8 concludes that $\mathbf{V} \twoheadrightarrow \mathrm{Q}$. Else, Q is one of those small queries that we cannot handle, but there are only finitely many of them. Hence α in Theorem 2 can be defined as the function that maps $\mathbf{V}$ to the maximal n given by Proposition 9. In order to decide general determinacy, we would need the *smallest* $n \equiv m[c]$ that is determined by $\mathbf{V}$. This particular issue is discussed in Section 5.

The proof of Proposition 9 is divided in three parts. In Section 4.1, we introduce a tool that describes the possible behaviors that can be observed through the view $\mathbf{V}$. We use it to

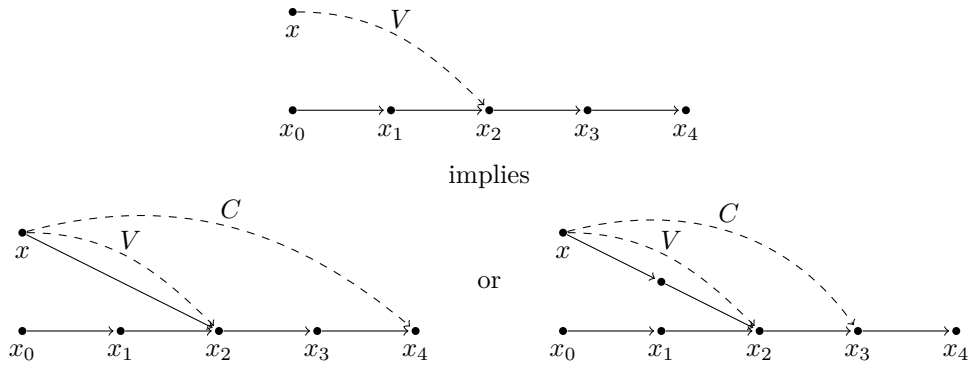

Figure 2 Example of possible behaviors for a database (full) and its view (dashed), with $C = \{3\}$ and $V = \{1, 2\}$. Assume we know the information represented in the top figure. Then, one of the two bottom pictures must hold. More generally, if $C = \{c\}$ and $V(x, x_i)$ holds in E, then $C^k(x, x_j)$ must also hold, with $j = i + (c - v[c])$ and $kc = v + (c - v[c])$ for some $v \in V$.

prove the propositions in Section 4.2 and Section 4.3. More precisely, Section 4.2 settles the case where no query $\mathrm{Q} = \{n\}$ with $n \equiv o[c]$ is determined by $\mathbf{V}$, and Section 4.3 builds an approriate query Q when one does exist.

4.1 Behavior graph

The goal of this section is to define a tool that will help us deal with the high combinatorial complexity when trying to find a target path in a database based on its view image. We now give a rough sketch of the idea behind this tool. Assume we want to prove that some database D contains a path π of length n by only looking at $\mathrm{E} = \mathbf{V}(\mathrm{D})$, where $\mathbf{V}$ is a C-complete view that contains only finite unions of path queries. If D does indeed contain such a path, then the following properties must necessarily hold in E:

C1. E contains the $n + 1$ (not necessarily distinct) nodes of π, $x_0, \ldots, x_n$.

C2. For each $V \in \mathbf{V}$ and $u \in V$, $V(x_i, x_{i+u})$ holds in E for all i.

C3. For each x in E such that $V(x, x_i)$ holds in E, there exists an appropriate value of k and j such that $C^k(x, x_j)$ holds in E. The values of k and j depend on the witness path that proves $V(x, x_i)$, as shown in Figure 2.

Of course, there are many ways for a view instance E to satisfy all these properties without D actually having a path of length n from x_0 to x_n, let alone one that goes through all the x_i's in the right order. Let μ be a path in D from some x_i to some x_j. We define the *delay* of this path as $\delta(\mu) = |\mu| - (j - i)$, that is the length of μ minus the expected length of μ if μ had been the section from x_i to x_j of a path of length n whose nodes are the x_i's. Note that $\delta(\mu)$ can be positive (μ is longer than expected), negative (μ is shorter than expected), or zero, in which case there is a path of length $(j - i)$ from x_i to x_j as intended.

Let D be a database and $\mathrm{E} = \mathbf{V}(\mathrm{D})$ such that E satisfies the necessary conditions above. For this D and E, we build a graph H_{D} that represents the delays of the paths of D that are induced by the conditions (C1), (C2) and (C3) as follows:

- H_{D} has $n + 1$ nodes that represent $x_0, \ldots, x_n$, as in (C1). We simply note them $0, \ldots, n$.
- For all $V \in \mathbf{V}$ and $u \in V$, (C2) implies that $V(x_i, x_{i+u})$ holds in E. Hence, there exists a path π in D going from x_i to x_{i+u} of length v, for some $v \in V$.
  - We represent this as an edge in H_{D} going from i to $i + u$ of label $\delta(\pi) = (v - u)$.

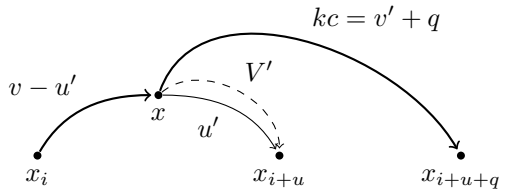

Figure 3 Illustration for the existence of the path π' of delay $(v-u)+(v'-u')$ in the construction of H_{D}. Full arrows represent paths in D and are labeled by their length. Dashed arrows represent edges in E. π' is the thick path, and $q = c - v'[c]$.

- For all $u' < v$ such that $u' \in V'$ for some $V' \in \mathbf{V}$, we know that $V'(x, x_{i+u})$ holds in E, where x is the u'th predecessor of x_{i+u} along π. We apply (C3) as shown in Figure 3. This leads to a path π' from x_i to $x_{i+u+(c-v'[c])}$ such that $\delta(\pi') = (v-u)+(v'-u')$, for some $v' \in V'$, which we similarly represent in H_{D}.

Assume that there is a path from node 0 to node n in H_{D} whose sum of labels is 0. By composing all the paths in D that led to this path in H_{D}, we can prove that there exists in D a path π from x_0 to x_n such that $\delta(\pi) = 0$. Hence, π is of length n, and we have actually found a path of length n from x_0 to x_n in D.

Consider the case where this is true for all databases D, that is, for all databases D such that $\mathbf{V}(\mathrm{D})$ satisfies the necessary conditions, H_{D} contains such a path. Then all these databases contain a path of length n from x_0 to x_n. This means that the necessary conditions for the existence of a path of length n in D are also sufficient. Since these conditions can be checked by looking only at the view instance, it implies that $\mathbf{V} \twoheadrightarrow \{n\}$.

Unfortunately, the size of H_{D} depends on the size of the target query. In order to have a representation that does not depend on n, we identify in H_{D} all nodes i and j such that $i \equiv j[c]$. Note that this is consistent with the fact that such nodes were already linked by paths of delay 0 thanks to $C \in \mathbf{V}$. This is exactly the idea behind *choice graphs*, that are formally defined below. While we do lose some information by doing this merging, these graphs are still rich enough to allow us to decide asymptotic determinacy, as we will see in the rest of the proof.

▶ **Definition 10** (Choice graph)**.** Given a C-complete view $\mathbf{V}$ such that $C \in \mathbf{V}$ with $C = \{c\}$, we define $\mathcal{H}_{\mathbf{V}}$ as the set of all directed, edge-labeled graphs H such that:

1. H has c nodes, which we will simply note $0, 1, \ldots, c-1$.
2. The edges of H carry labels in $\{-2(m-1), \ldots, 2(m-1)\}$, where m is the biggest element that appears in the views, that is $m = \max_{V \in \mathbf{V}} \max_{u \in V} u$.
3. For each $i, j \in \{0, \ldots, c-1\}$, for each $V \in \mathbf{V}$, for each $u \in V$ such that $u \equiv (j-i)[c]$, there exists $v \in V$ such that:
   - there is an edge in H from i to j labeled by $v - u$.
   - for each $V' \in \mathbf{V}$, for each $u' \in V'$, there exist $v' \in V'$ and an edge in H from i to $(j - v')[c]$ labeled by $(v-u)+(v'-u')$.

▶ Remark. Since each $H \in \mathcal{H}_{\mathbf{V}}$ has a bounded number of nodes and edges, then $\mathcal{H}_{\mathbf{V}}$ is finite. Moreover all $H \in \mathcal{H}_{\mathbf{V}}$ are complete graphs, because $\mathbf{V}$ is C-complete.

▶ **Definition 11** (Weight)**.** The weight of a path in a graph H is the sum of all labels along edges of the path. A path with no edge is of weight 0.

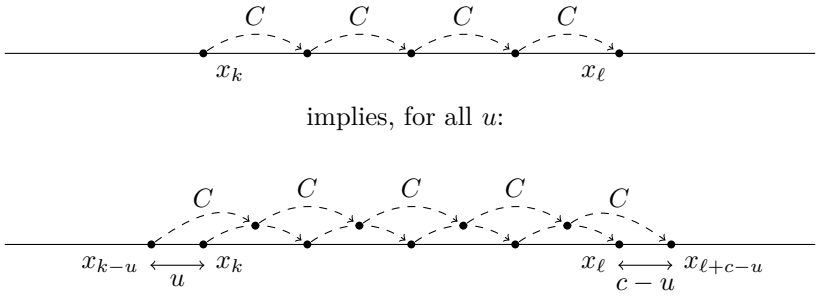

Figure 4 Illustration of the intuition for the construction of a behavior graph. Assume that the x_i's form a path of length n from x_0 to x_n. If x_k and x_ℓ are connected via a sequence of C's, represented by the dashed edges, then for all $u < c$, there exists some intermediate nodes such that x_{k-u} and $x_{\ell+c-u}$ are connected as shown in the picture.

▶ **Definition 12** (Behavior graph)**.** Given a C-complete view $\mathbf{V}$ such that $C \in \mathbf{V}$ with $C = \{c\}$, we define $\mathcal{G}_\mathbf{V}$ as the set of all directed, edge-labeled graphs G constructed as follows:

1. Pick $H \in \mathcal{H}_\mathbf{V}$, and start with $G = H$.
2. Pick $i, j \in \{0, \dots, c-1\}$ such that:
   - There exists in G a path from i to j of weight $(i-j)[c]$. Let a be the weight of a path of minimal length satisfying this property.
   - For all $a' \equiv a[c]$, there exists i', j' such that $(j'-i') \equiv (j-i)[c]$, and there is no edge from i' to j' of label a'.

   Then, for all i', j' such that $(j'-i') \equiv (j-i)[c]$, add an edge a from i' to j'.
3. Repeat step 2 until no more edges can be added.

▶ Remark.

- Step 2 of the construction of $\mathcal{G}_\mathbf{V}$ can only be applied a finite number of times for each G, since it can be done at most once for each (i, j). Moreover, there is a finite amount of choice at each step. Hence, $\mathcal{G}_\mathbf{V}$ is finite.
- As soon as there is a path from some i to some j of weight $(i-j)[c]$, then there is a weight $a \equiv (i-j)[c]$ such that all i', j' that are at the same distance than i is from j are linked by an edge of this particular weight.

Behavior graphs contain another necessary property of the existence of a path of length n that goes through the x_i's. Remark that a path π of delay $i-j$ going from x_i to x_j is actually of length 0 modulo c. Hence π appears in E as a sequence of C edges. Then the reasoning shown in Figure 4 implies the existence of paths of identical delay from nodes x_{i-u} to nodes x_{j+c-u} for all u.

All the intuitions presented in this section are made precise in Section 4.2 and Section 4.3, when this tool is actually used. In Section 4.2, we show that, if there exists some behavior graph G such that there is no path of weight 0 from 0 to o in G, then we can build two databases that agree on the view but not on paths of length $n \equiv o[c]$, for all n. In other words, we can build a database whose view satisfies all the necessary conditions for the existence of a path of length n, while still maintaining a non-zero delay between the relevant nodes. On the contrary, in Section 4.3, we show that, if for all behavior graphs G, there is a path of weight 0 from 0 to o in G, then it is enough to satisfy the necessary conditions in order to have a path of length n.

Our decision algorithm uses these properties of behavior graphs as follows. For a given C-complete view $\mathbf{V}$ and a given natural number $o \in \{0, \dots, c-1\}$, we are simply looking

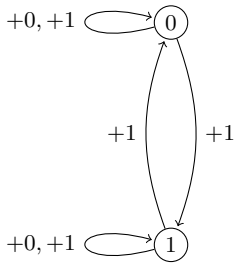

Figure 5 A behavior graph for the view defined in Example 14.

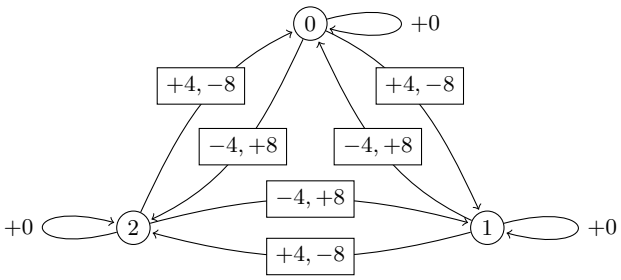

Figure 6 A behavior graph for the view defined in Example 15.

for the occurrence of a specific graph $G \in \mathcal{G}_{\mathbf{V}}$, namely one that does not contain a path of weight 0 from node 0 to node o. If we do find one such G, then, for all $n \equiv o[c]$, $\mathbf{V} \not\twoheadrightarrow \{n\}$. Otherwise, $\mathbf{V} \twoheadrightarrow \{n\}$ for some $n \equiv o[c]$. We do not actually need to compute $\mathcal{G}_{\mathbf{V}}$: we simply guess the appropriate graph G and check that it does contain the critical path. Since G is of size polynomial in $\mathbf{V}$ and the considered path, if it exists, can be assumed to be polynomial in the size of G (thanks to Bezout's Identity), our decision procedure is in PSPACE, more precisely in Π_2^{P}.

4.2 Negative direction: building counter-examples

In this section, we solve the negative case of Proposition 9 by proving the following proposition:

▶ **Proposition 13.** *Assume that there exists $G \in \mathcal{G}_{\boldsymbol{V}}$ such that there is no path of weight 0 from 0 to o. Then, for all $n \equiv o[c]$, $\boldsymbol{V} \not\twoheadrightarrow \{n\}$.*

The proof of Proposition 13 is split across Lemma 16 and Lemma 17. The canonical counter-examples that we build in order to prove that $\mathbf{V} \not\twoheadrightarrow \{n\}$ depend on whether G only contains cycles of positive or negative weights, or if it actually has both. These two cases are respectively dealt with in Lemma 16 and Lemma 17, and Example 14 and Example 15 give examples of both situations.

▶ **Example 14.** Let $\mathbf{V} = \{C, V\}$, $C = \{2\}$ and $V = \{1, 2\}$. Figure 5 represents one of the graphs in $\mathcal{G}_{\mathbf{V}}$, that additionally satisfies the condition of Lemma 16. Namely, there is no path of weight 0 from 0 to 1, and 0 only has non-negative cycles.

▶ **Example 15.** Let $\mathbf{V} = \{C, V\}$, $C = \{3\}$ and $V = \{1, 5\}$. Figure 6 represents one of the graphs in $\mathcal{G}_{\mathbf{V}}$, that additionally satisfies the condition of Lemma 17. Namely, there is no path of weight 0 from 0 to 1, and 0 has positive and negative cycles.

▶ **Lemma 16.** *Assume that there exists $G \in \mathcal{G}_{\boldsymbol{V}}$ such that 0 does not have both a cycle of positive weight and a cycle of negative weight and that there is no path of weight 0 from 0 to o. Then, for all $n \equiv o[c]$, $\boldsymbol{V} \not\twoheadrightarrow \{n\}$.*

Sketch of proof. Assume that G does not have cycles of negative weights. An example of such a case is given in Example 14. We explain how the proof works on this example.

We build a counter-example showing that $\mathbf{V}$ does not determine any odd n as follows. Start from a database D which consists of two simple paths of length n denoted $x_0 \dots x_n$ and $x'_0 \dots x'_n$. For each i, add a path of length 2 from x_i to x_{i+1}, and a path of length 2 from x'_i to x'_{i+1}, sharing their middle nodes. Notice now that you can switch the position of

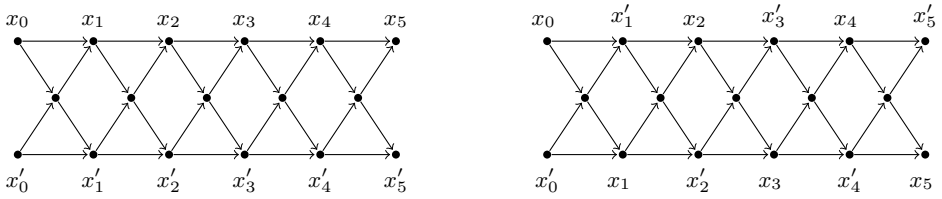

Figure 7 Example of the construction in Lemma 16 for the view defined in Example 14.

x_i with x'_i for all odd i without altering the view. This defines a new database D′, as shown in Figure 7. D and D′ agree on the view, but any path that goes from x_0 to x_n in D′ has to go at least once through one of the new paths, and is thus longer than n. Hence D and D′ disagree on Q, which concludes the proof.

This construction highlights the ideas in the general proof, for any $\mathbf{V}$ and Q. The fact that the new paths introduced from x_0 to x'_n in D are strictly longer than n actually stems from the assumption that G has no cycles of negative weights. The complete proof consists in giving an exact characterization of which new paths to add, and which nodes to switch between the two original paths. ◀

▶ **Lemma 17.** *Assume that there exists $G \in \mathcal{G}_{\mathbf{V}}$ such that 0 has both cycles of positive and negative weight, and that there is no path of weight 0 from 0 to o. Then, for all $n \equiv o[c]$, $\mathbf{V} \not\twoheadrightarrow \{n\}$.*

Sketch of proof. Remark that the conditions of this lemma are satisfied in Example 15, for $o = 1$. Let $\mathrm{Q} = \{n\}$, with $n \equiv 1[c]$. We illustrate the proof on this particular example.

Let W be the set of all cycles of 0 in G. Let $d = \gcd(W)$. Here, $d = 12$. Notice that, for each $i, j \in G$, there is a unique $w(i,j) \in \{0, \ldots, 11\}$ such that, for all path π from i to j, π is of weight $w(i,j)$ modulo d. For instance, all paths from 0 to 1 are of weight 4 modulo 12. A careful analysis of the arithmetic properties of G can actually show that this is true in general.

We build a counter-example showing that $\mathbf{V} \not\twoheadrightarrow \mathrm{Q}$ by using the information contained in G as follows. Start with a database D which is a cycle of length d whose nodes are $x_0, \ldots, x_{d-1}$. From D, we define a database D′ by renaming each node x_i of D to $x_{i+w(0,i[c])}$. For instance, x_0 remains as x_0, x_1 is renamed to $x_{1+w(0,1)}$, which is x_5, x_2 is renamed to x_{10}, and so on. See Figure 8.

Remark now that D′ is also a cycle of length d, whose nodes have simply been reordered, compared to D. Moreover, we can check that $\mathbf{V}(\mathrm{D}) = \mathbf{V}(\mathrm{D}')$. However, all paths of D of length 1 modulo c starting from x_0 end in one of x_1, x_4, x_7 or x_{10}, whereas for D′, they end in one of x_2, x_5, x_8 or x_{11}. This is due to the fact that $w(0,1) \neq 0$, as assumed by the lemma. Hence, D and D′ cannot agree on Q, which concludes this counter-example.

The fact that D′ is always a correctly-defined cycle of length d whose view image is equal to $\mathbf{V}(\mathrm{D})$ relies once again on the good arithmetic properties of G. ◀

4.3 Positive direction: computing a rewriting

In this section, we solve the positive case of Proposition 9. We start by giving a simple example that shows some of the features of the rewritings that will be used to prove Proposition 19.

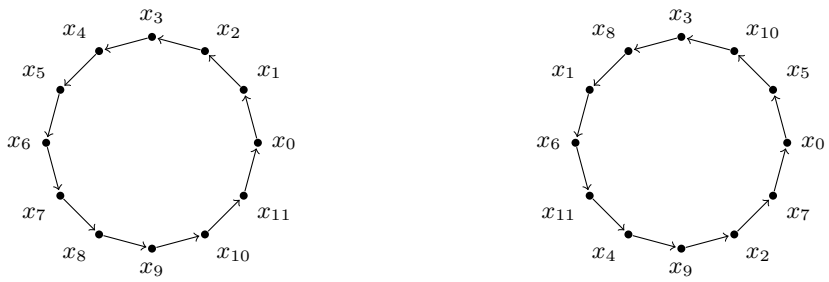

Figure 8 Example of the construction in Lemma 17 for the view defined in Example 15.

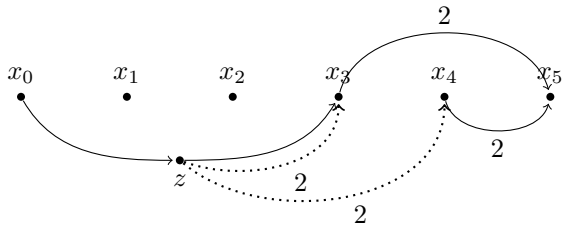

Figure 9 Illustration for the last case of Example 18. The full edges represent paths in the database, along with their length when it is more than 1. The dotted edges represent the two possible implications of $V_1(z, x_3)$.

▶ **Example 18.** In this example, we work with:

- $\mathbf{V} = \{C, V_1, V_2\}$
- $C = \{2\}$
- $V_1 = \{1, 2\}$
- $V_2 = \{2, 3\}$
- $\mathrm{Q} = \{5\}$

We show that $\mathbf{V} \twoheadrightarrow \mathrm{Q}$. Indeed, the following formula R is a rewriting of Q using $\mathbf{V}$.

$$\mathrm{R}(x, y) = \exists x_0, \ldots, x_5,\ x_0 = x \wedge x_5 = y \wedge CQ_{\pi_5} \wedge \Big(\forall z, V_1(z, x_3) \Rightarrow (C(z, x_3) \vee C(z, x_4))\Big)$$

where π_5 is a simple path whose nodes are $x_0, \ldots, x_5$ and CQ_{π_5} is the conjunctive query that states all the atoms that hold in $\mathbf{V}(\pi_5)$. First, remark that R only states necessary conditions for the existence of a path of length 5 from x to y, as explained in Section 4.1, hence, for all D, $\mathrm{Q}(\mathrm{D}) \subseteq \mathrm{R}(\mathbf{V}(\mathrm{D}))$.

Assume now that $(x, y) \in \mathrm{R}(\mathbf{V}(\mathrm{D}))$. Let $x_0, \ldots, x_5$ be a quantification for which $\mathrm{R}(x, y)$ is satisfied. We can prove the following:

- $C(x_0, x_2)$, $C(x_1, x_3)$ and $C(x_2, x_4)$ hold in $\mathbf{V}(\mathrm{D})$. Hence, these pairs of nodes are at distance 2 in D.
- $V_1(x_4, x_5)$ holds in $\mathbf{V}(\mathrm{D})$. Hence, x_4 and x_5 are either at distance 1 or 2. If this distance is 1, then we immediately get a path of length 5 from x_0 to x_5 by using the previous point, as $x_0 \xrightarrow{2} x_2 \xrightarrow{2} x_4 \xrightarrow{1} x_5$.
- Similarly, $V_2(x_0, x_3)$ holds in $\mathbf{V}(\mathrm{D})$. If the distance from x_0 to x_3 is 3, we immediately get $x_0 \xrightarrow{3} x_3 \xrightarrow{2} x_5$. Otherwise, there exists z such that $x_0 \to z \to x_3$. This implies $V_1(z, x_3)$.
- The remaining case is represented in Figure 9, with the two possible implications of $V_1(z, x_3)$ given by R. Both possibilities also imply a path of length 5 from x_0 to x_5.

▶ **Proposition 19.** *Assume that for all $G \in \mathcal{G}_{\mathbf{V}}$ there is a path of weight 0 from 0 to o. Then there exists $n \equiv o[c]$ such that $\mathbf{V} \twoheadrightarrow \{n\}$ and we can effectively compute a first-order rewriting that witnesses it.*

Sketch of proof. Let $\mathrm{Q} = \{n\}$, with $n \equiv o[c]$. We define a rewriting R as:

$$\mathrm{R}(x,y) = \exists x_0, \ldots, x_n,\ x_0 = x \wedge x_n = y \wedge \bigwedge_{i=1}^{3} R_i(x_0, \ldots, x_n)$$

where R_1, R_2 and R_3 are first-order formulas such that:

- R_1 states that $x_0, \ldots, x_n$ satisfy $\mathbf{V}(\pi)$ with $\pi = x_0 \ldots x_n$.
- R_2 states that, if $V(z, x_i)$ holds for some z, then $C^{\lfloor u/c \rfloor + 1}(z, x_{i+c-u[c]})$ must also hold for some $u \in V$, as explained in Figure 2.
- R_3 states that, if x_i and x_j are at distance $d \equiv 0[c]$, then x_{i-l} and x_{i+c-l} must be at distance $d + c$, for all $l \in \{0, \ldots, c-1\}$. As stated, R_3 is not in first-order because it is an infinite disjunction on the values of d. We can actually make it finite, but we omit the argument here for simplicity.

First, remark that $\mathrm{Q}(\mathrm{D}) \subseteq \mathrm{R}(\mathbf{V}(\mathrm{D}))$. Indeed, each R_i only states necessary conditions. The rest of this sketch is devoted to showing the converse.

Let D be a database such that $(x, y) \in \mathrm{R}(\mathbf{V}(\mathrm{D}))$. Hence, there exist nodes $x_0, \ldots, x_n$ in D such that $\mathrm{R}_i(x_0, \ldots, x_n)$ holds in $\mathbf{V}(\mathrm{D})$. Let π be a path in D from x_i to x_j, we denote the delay of this path by $\delta(\pi) = |\pi| - (j - i)$, that is the length of π minus the expected distance between x_i and x_j. For instance, assume that there is a path of length 2 from x_0 to x_1, then the delay of this path would be 1. By following this path, x_1 is too far from x_0 by 1, because x_1 was intended to be the successor of x_0.

From D, we build a graph H with $n + 1$ nodes such that there is an edge of label w between nodes i and j if and only if there is a path π from x_i to x_j with $\delta(\pi) = w$. Then we deduce from H a graph G by merging the nodes of H that have the same index modulo c. Remark that R_1, R_2 and R_3 imply that G actually contains a behavior graph G_0, as defined in Section 4.1. Thus, by hypothesis, this graph contains a path π_0 from 0 to o of weigth 0.

If n is big enough, then we can actually choose G_0 in such a way that each edge of G_0 has many witness paths in D. In other words, if G_0 contains an edge from i to j of weight w, then there exists many pairs of nodes $x_{i'}$ and $x_{j'}$ such that $i' \equiv i[c]$, $j' \equiv j[c]$ and there exists a path π from $x_{i'}$ to $x_{j'}$ with $\delta(\pi) = w$. With some additional combinatorial work, we show that we can mimic π_0 in D through the use of these witness paths, and thus produce a path of delay 0 from x_0 to x_n. Hence, this path is of length n, which implies that $(x_0, x_n) \in \mathrm{Q}(\mathrm{D})$ and concludes the proof. ◀

5 The case of small queries

In this section, we investigate the following example, in order to illustrate the issue that remains to be solved in the case of small queries.

- $\mathbf{V} = \{C, V_1, V_2\}$
- $C = \{2\}$
- $V_1 = \{1, 2\}$
- $V_2 = \{2, 5\}$

▶ **Claim 20.** *For all even n, $\boldsymbol{V} \twoheadrightarrow Q = \{n\}$. This easily comes from $C = \{2\}$.*

By applying Theorem 9 we can show that there exists some odd n such that $\mathbf{V} \twoheadrightarrow \mathrm{Q} = \{n\}$, hence $\mathbf{V}$ also determines all bigger queries. In order to get the full picture, we need to find the smallest odd n that is determined by $\mathbf{V}$. Our work so far actually gives us:

▶ **Claim 21.** *For all odd $n \leq 7$, $\boldsymbol{V} \not\twoheadrightarrow Q = \{n\}$.*

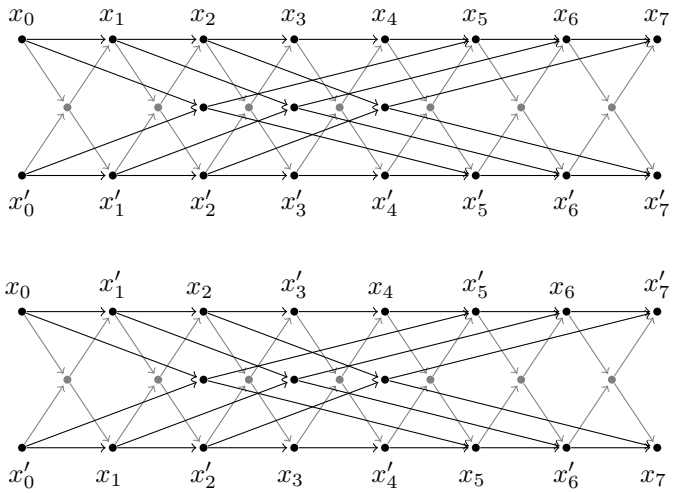

Figure 10 The two databases above are a proof that $\mathbf{V} \not\twoheadrightarrow \mathrm{Q} = \{n\}$ for any odd n that is not greater than 7. Indeed, we can check that both databases agree on $\mathbf{V}$. However, there is no path of length 1 (respectively 3, 5 and 7) from x_0 to x_1 (respectively x_3, x_5 and x_7) in the bottom database.

To prove this claim, we use a technique that is very similar to Lemma 16. More precisely, the two databases in Figure 10 agree on $\mathbf{V}$, but disagree on all $\mathrm{Q} = \{n\}$ when n is odd and not greater than 7.

Note that this technique does not work for n greater than 7. Indeed, in the case shown above, any path that goes from x_0 to x_7 in the bottom database has to cross from the top section to the bottom section. By doing so, it suffers a delay of either $+1$ or -3 compared to the expected value. It works here because 7 is "too small" and does not provide enough space to catch-up on this delay. Assume now that $n = 9$, then a delay of -3 can be mitigated by following a $+1$ path three times, and thus does not provide a counter-example.

▶ **Claim 22.** *For all $n \geq 11$, $\boldsymbol{V} \twoheadrightarrow Q = \{n\}$.*

We show this by arguing that $\mathbf{V} \twoheadrightarrow \mathrm{Q} = \{11\}$. This is done by actually proving that the canonical rewriting R given in Section 4.3 works in this case. Although the proof given in Section 4.3 does not apply (because 11 is not "big enough" for all the combinatorial arguments to go through), a careful enumeration of all the possibilities for a database satisfying R actually shows that $\mathrm{R}(x, y)$ implies a path of length 11 from x to y, as was done in Example 18.

It is then straightforward to prove that $\mathbf{V}$ determines every odd query bigger than 11. Let $n = 11 + 2k$ be such a query. Then a rewriting for n is simply $\mathrm{R}_{11} \cdot C^k$, as in Lemma 8. As we already know that $\mathbf{V}$ determines every even query, this end the proof of the claim.

The case of $n = 9$. There remains only a single unsolved case, which is $n = 9$. This qualifies as a "small query" for the view $\mathbf{V}$: a query for which we are unable to either build a generic counter-example, as in Section 4.2, or provide a generic rewriting, as in Section 4.3. We actually proved that $\mathbf{V} \not\twoheadrightarrow \mathrm{Q} = \{9\}$. However, the smallest counter-example that we know is a pair of databases of 154 nodes each, that were built by hand through a very tedious trial and error process and checked by a computer program. At this time, we are unfortunately unable to provide any technique to generate such a counter-example for other views and queries. We conjecture that the combinatorial complexity of these "small queries" might be way higher than what we have dealt with so far.

6 Conclusions

We have shown that, given a view $\mathbf{V}$ defined by unions of path queries, we can decide determinacy of almost all path queries Q. Although the smallest query that we can handle is of exponential size in the size of $\mathbf{V}$, our decision procedure still works with Π_2^{P} complexity. Moreover, for the queries that are big enough to be handled by our algorithm, we also provide a first-order rewriting when they are determined, and a canonical counter-example otherwise.

A natural continuation of this work would be to try and solve the determinacy problem even for small queries. Another possible continuation stems from the following remark: on all examples where $\mathbf{V}$ determines Q that we are aware of, it also turns out that our rewriting is actually correct, even when the query is too small to be handled by our technique. Perhaps it so happens that this rewriting is always correct, as soon as we assume that $\mathbf{V}$ determines Q. Failing that, it might still be the case that a first-order rewriting can always be found.

Acknowledgements. I gratefully thank Luc Segoufin and Cristina Sirangelo for carefully proofreading this paper and providing many invaluable comments and advices.

References

1 Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *ACM Symp. on Principles of Database Systems (PODS)*, pages 254–263, 1998.

2 Foto N. Afrati. Determinacy and query rewriting for conjunctive queries and views. *Theoretical Computer Science*, 412(11):1005–1021, 2011.

3 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Lossless regular views. In *ACM Symp. on Principles of Database Systems (PODS)*, pages 247–258. ACM, 2002.

4 Nadime Francis, Luc Segoufin, and Cristina Sirangelo. Datalog rewritings of regular path queries using views. In *Proceedings of the 17th International Conference on Database Theory (ICDT'14)*, pages 107–118, 2014.

5 Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *ACM Symp. on Principles of Database Systems (PODS)*, pages 95–104, 1995.

6 Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Transactions on Database Systems*, 35(3), 2010.